UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/072,358	02/06/2002	Kenneth C. Duisenberg	10019681-1	4285

22879 7590 01/19/2010
HEWLETT-PACKARD COMPANY
Intellectual Property Administration
3404 E. Harmony Road
Mail Stop 35
FORT COLLINS, CO 80528

EXAMINER	
LEE, CHUN KUAN	

ART UNIT	PAPER NUMBER
2181	

NOTIFICATION DATE	DELIVERY MODE
01/19/2010	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

JERRY.SHORMA@HP.COM
ipa.mail@hp.com
laura.m.clark@hp.com

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Ex parte KENNTH C. DUISENBERG

Appeal 2009-002167
Application 10/072,358[1]
Technology Center 2100

Decided: January 14, 2010

Before LEE E. BARRETT, JOSEPH L. DIXON, and THU A. DANG, *Administrative Patent Judges.*

BARRETT, *Administrative Patent Judge.*

DECISION ON APPEAL

This is a decision on appeal under 35 U.S.C. § 134(a) from the final rejection of claims 1-24. We have jurisdiction pursuant to 35 U.S.C. § 6(b).

We reverse.

---

[1] Filed February 6, 2002, titled "Method and Apparatus for Synchronizing a Software Buffer Index with an Unknown Hardware Buffer Index." The real party in interest is Hewlett-Packard Development Company, L.P.

STATEMENT OF THE CASE

The invention

The invention relates to a method and apparatus for synchronizing a software buffer index with an unknown hardware buffer index. As described in the background prior art, a local area network (LAN) software driver works in conjunction with the LAN hardware to process data stored in memory by the LAN hardware. As shown in Figure 1, data is stored and accessed from a ring of buffers. Both the LAN hardware and the LAN software have respective buffer indices that indicate which buffer in the ring of buffers the corresponding hardware or software will access. With normal data processing when the buffer indices are synchronized, the LAN software would receive an interrupt from the LAN hardware that a new LAN packet of data was received and ready for processing. The LAN software would check its buffer index and find the new data to process. Spec. 1, ll. 5-22.

The LAN hardware buffer index and the LAN software buffer index may become unsynchronized. If the LAN hardware buffer index is not pointing to the same memory location as the LAN software buffer index when the LAN software receives an interrupt indicating a LAN packet containing data has been stored and is ready for processing, the LAN software will check the buffer indicated by its buffer index and discover that there is nothing new to process. In that case, the LAN software will wait without advancing its buffer index. Meanwhile, the LAN hardware will continue to receive incoming LAN packets of data and store them into succeeding buffers having already processed data. Spec. 2, ll. 11-19.

The LAN hardware and software buffer indices will continue to be unsynchronized until the LAN hardware buffer index has moved to the same location as the LAN software index. The LAN software buffer index would be synchronized with the LAN hardware buffer index until the LAN software buffer index reaches all the previously stored buffers in the ring still containing unprocessed data. At that point, the LAN software would process all the unprocessed data within a single time period, advancing the software index until reaching a buffer containing processed data. As such, the LAN software buffer index would again be out of synchronization with the LAN hardware buffer index. Spec. 2, ll. 21-30.

One problem with letting the buffer indices remain unsynchronized would be an inefficient use of processing power since the processor would remain idle for a period, and then would have to catch up by processing all the queued buffers containing unprocessed data. Spec. 5.

The invention synchronizes the software buffer index with the hardware buffer index by sequentially searching through a plurality of buffers containing data to find a buffer with unprocessed data. The method is implemented when the software buffer index points to a first buffer containing processed data. Thereafter, the software buffer index is reset to the next available buffer having processed data following the second buffer.

Illustrative claim

Claim 1 is reproduced below for illustration:

1. A method of processing data comprising:

when a software buffer index points to a first buffer containing processed data, synchronizing said software buffer index to a hardware buffer index by sequentially searching through a plurality of buffers containing data to determine whether there is a second buffer with unprocessed data; and

if there is said second buffer with unprocessed data, resetting said software buffer index to a next available buffer having processed data following said second buffer, and otherwise stopping said searching when each buffer of said plurality of buffers has been searched and a buffer with unprocessed data is not found.

The references

The Examiner relies on the following prior art:

Lounsbury	4,637,023	Jan. 13, 1987
Cromer	5,860,001	Jan. 12, 1999

Applicant's Admitted Prior Art (AAPA) described at Specification 1-5 and Figure 1.

The rejections

Claims 1, 2, 4-17, and 19-24 stand rejected under 35 U.S.C. § 103(a) as unpatentable over AAPA and Lounsbury.

Claims 3 and 18 stand rejected under 35 U.S.C. § 103(a) as unpatentable over AAPA and Lounsbury, further in view of Cromer.

FINDING OF FACT

Content of the references

AAPA

The AAPA is described in the description of the invention.

Lounsbury

Lounsbury relates to correcting recording errors in writing to and reading from a serially recorded magnetic recording tape. Col. 1, ll. 7-10. A problem with prior art magnetic tape controllers is that they have been unable to correct recording errors "on-the-fly." One reason is that most prior art controllers have been able to work only with complete magnetic tape data records. Col. 4, ll. 28-38.

Lounsbury describes a solution which involves breaking a conventional data record into a plurality of smaller data pieces called "blockettes." Each blockette is assigned a unique sequential number, which is recorded immediately prior to the data in the blockettes. The number is used to assemble the blockettes in sequential order to reconstruct the original data record. Col. 4, ll. 39-46.

Data to be recorded on a tape and to be read from the tape to a host computer is stored in a buffer ring. As shown in Figure 6, the ring has four buffers. Each buffer stores a blockette having a blockette number 508 and cyclic redundancy code (CRC) 512, 514. Col. 13, l. 42 to col. 14, l. 25.

When writing data to the tape, data is read shortly after it is written to the tape to determine whether it has been recorded properly. Since tape writing of the next sequential block is started before the read operation of the preceding block is finished, blockettes may be written containing errors. However, the blockettes with errors are rewritten and the blockette numbers are used to put the blockettes back in order. Col. 9, ll. 31-58.

The first blockette 1 (element 220) is written to the tape as shown in the lower row of blocks in Figure 2. Shortly thereafter, the blockette is read as shown at the upper row of blocks and the CRC is calculated and compared to the CRC stored on the tape. If the CRCs match, there is no error and the acknowledge count is incremented allowing the next sequential blockette to be written to the tape. Col. 11, ll. 14-20. If the CRCs do not match, there is an error; the acknowledge count is decremented and the blockette with the error is rewritten. Col. 11, l. 47 to col. 12, l. 35.

There are two read routines for reading data from the tape to the host computer. The first routine reads data from the tape and writes it to the buffers. The second routine reads data from the buffers and transfers it to the host computer system. Col. 12, l. 67 to col. 13, l. 2.

The tape-to-buffer read routine looks for an empty buffer and writes data from the tape into the buffer. The computed CRC is checked against the stored CRC code and if no error is found, the blockette data is marked "valid" and the procedure is repeated to write the next blockette into an empty buffer. If the CRC does not match, the data is marked "invalid" and the procedure repeats to write the next blockette. Col. 13, ll. 3-17.

The second read routine checks buffers in sequence for valid data. If a buffer does not contain valid data the next buffer in the chain is checked.

If a buffer contains valid data, the blockette number is checked to see if it is the next blockette in the blockette number sequence. If it is, that blockette is transferred to the host computer. If it is not, the routine scans the buffers for the next blockette in the sequence. Col. 13, ll. 18-29. If all four buffers (the entire ring) are scanned without finding a matching blockette number, an error condition is indicated. Col. 18, ll. 41-48.

Differences between claim 1 and AAPA

The AAPA teaches a ring buffer having a hardware buffer index and a software buffer index, but does not teach the following differences.

First, AAPA does not teach "when a software buffer index points to a first buffer containing processed data, . . . sequentially searching through a plurality of buffers containing data to determine whether there is a second buffer with unprocessed data" because if the software index points to a buffer containing processed data, the LAN software in the AAPA will wait without advancing its buffer index, Spec. 2, ll. 11-17, and therefore does not search for a buffer with unprocessed data.

Second, as to the limitation "if there is said second buffer with unprocessed data, resetting said software buffer index to a next available buffer having processed data following said second buffer," the AAPA describes resetting the software buffer index to a next available buffer having processed data after processing unprocessed data in a buffer. For example, after processing unprocessed data in buffer N in Figure 1, the software buffer index is reset to buffer 0 which contains processed data.

Spec. 4, ll. 20-33. However, AAPA does not perform this step after having first searched for a buffer with unprocessed data.

Third, the AAPA does not teach "otherwise stopping said searching when each buffer of said plurality of buffers has been searched and a buffer with unprocessed data is not found" because it does not teach searching the plurality of buffers for a buffer with unprocessed data.

ISSUE

Has Appellant shown that the Examiner erred in concluding that Lounsbury would have taught or suggested modifying the AAPA to provide "synchronizing said software buffer index to a hardware buffer index" by sequentially searching the buffers and resetting the software buffer index to the next available buffer having processed data as recited in claim 1?

PRINCIPLES OF LAW

"[T]he test [for obviousness] is what the combined teachings of the references would have suggested to those of ordinary skill in the art." *In re Keller*, 642 F.2d 413, 425 (CCPA 1981). A rejection under 35 U.S.C. § 103(a) is based on the following factual determinations: (1) the scope and content of the prior art; (2) the level of ordinary skill in the art; (3) the differences between the claimed invention and the prior art; and (4) any objective indicia of non-obviousness. *KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398, 399 (2007) (citing *Graham v. John Deere Co.*, 383 U.S. 1, 17-18 (1966)). All claim limitations must be taught or suggested.

ANALYSIS

The AAPA teaches a ring buffer having a hardware buffer index and a software buffer index but does not teach the steps for "synchronizing said software buffer index to a hardware buffer index" as noted in the findings of the differences between the AAPA and the subject matter of claim 1.

Lounsbury does not describe the problem of synchronizing a software buffer index with an unknown hardware buffer index. Lounsbury does not check for processed or unprocessed data to cause the software buffer index to change. Thus, Lounsbury is different from the claimed invention. However, the Examiner concludes that Lounsbury describes an analogous ring buffer read/write technique which one of ordinary skill in the art would have recognized as teaching a solution to the problem in the AAPA of synchronizing a software buffer index with a hardware buffer index. In particular, the Examiner finds that Lounsbury teaches:

> when an error occurs (e.g. existence of invalid data as there is unprocessed data in said second buffer when the software buffer index is pointing to the current first buffer contain processed data) while transferring the data from the ring of buffers to the host computer, the ring of buffers is sequentially searched for the correct data to continue transferring, and stop the searching if the correct data is not found after searching all the buffers in the ring of buffers (col. 13, ll.18-41 and col. 18, ll. 41-52); therefore, if there is said second buffer with unprocessed data when the software buffer index is pointing to the current first buffer contain processed data (e.g. error associated with existence of invalid data in the ring buffer), resetting said software buffer index to a next available buffer having processed data following said second buffer by sequential searching to find and correct the error, and after the error is corrected by processing the unprocessed data in said second buffer, said software buffer index

> would be reset to the next available buffer having processed data following said second buffer.

Ans. 6.[2] That is. the Examiner relies on Lounsbury's description of "transferring data from the ring of buffers to the host computer" (Ans. 6) and finds that Lounsbury's "valid data" corresponds to the claimed "processed data" and Lounsbury's "invalid data" corresponds to the claimed "unprocessed data" and Lounsbury's teaching of scanning the buffers at column 13, lines 18-41 and column 18, lines 41-54 corresponds to the claimed step of "sequentially searching through a plurality of buffers containing data to determine whether there is a second buffer with unprocessed data." Ans. 13.

Initially, we note that column 18, lines 41-54 describes writing data from the buffers to the tape, not transferring data from the buffers to the host computer. Thus, we consider only the description of transferring data from the buffers to the host computer as stated in the rejection.

Appellant argues that Lounsbury does not mention a buffer index. Br. 11. It is true that Lounsbury does not describe a "hardware buffer index" and a "software buffer index." However, Lounsbury has a first read routine that reads data from the tape and stores it in a buffer (col. 13, ll. 3-17), and it must have an index pointer to identify the buffer--this is effectively a "write index" which is equivalent in function to the claimed "hardware buffer

[2] We refer to the second Examiner's Answer entered April 9, 2008, the Brief filed January 28, 2008, and the Reply Brief filed May 19, 2008.

index" which points to a buffer to write to. Lounsbury has a second read routine that transfers data from the buffers to the host computer device, and it must have an index pointer to identify the buffer that is being read--this is effectively a "read index" which is equivalent in function to the "software buffer index" which points to a buffer to read from. Lounsbury describes that CUR_PTR contains the address of the buffer next in the buffer chain to the buffer which is involved in a data transfer (either into or out of the buffer) and CUR_PDMA which contains the address of the buffer which is currently being transferred to the host computer system. Col. 14, ll. 37-44. An address or pointer is considered to be an "index."

Appellant argues that Lounsbury's checking for valid data is not analogous to checking for processed or unprocessed data and would not have suggested the claimed invention. Br. 10-11; Reply Br. 2, 4. We have considered the Examiner's reasoning but agree with Appellant that Lounsbury would not have taught or suggested the claimed invention. The only things Lounsbury and the claims have in common is that both use a buffer ring to buffer data as it is being read and both scan a buffer ring: Lounsbury scans the buffer ring for a buffer with valid data (in reading) and for an empty buffer (when writing to the buffer), whereas the claims scan the buffer ring for a buffer with unprocessed data. The mere act of scanning a buffer ring for a certain condition does not suggest a solution to Appellant's claimed process of "synchronizing said software buffer index to a hardware buffer index" in buffers containing processed and unprocessed data by the claimed steps. Since Lounsbury does not check for processed or unprocessed data, there is no straightforward or apparent reason why one skilled in the art would have looked to or been motivated to apply the teachings of checking for valid data in Lounsbury to the AAPA.

Furthermore, it is not apparent how Lounsbury would have suggested modifying the AAPA to arrive at the claimed invention. Representative claim 1 recites "synchronizing said software buffer index to a hardware buffer index" by sequentially searching the buffers and resetting the software buffer index to the next available buffer having processed data. Although the action of the hardware buffer index is not described in claim 1, implicitly, the steps of claims 1 result in the software buffer index and hardware buffer index being synchronized. The Examiner has not shown, nor do we find, where Lounsbury describes that the read index will ever be synchronized with the write index. Lounsbury describes the writing of data from the tape to the buffers and reading of data from the buffers to the host computers as two separate processes. Lounsbury does not describe that the read process stops and points at the next buffer containing valid data, wherein the write index pointer is the same as the read index pointer and therefore does not suggest a solution to the problem in the AAPA. While Lounsbury scans the buffer ring when writing to and reading from the buffers, and while claim 1 recites a scan for the next buffer with unprocessed data, Lounsbury scans over full buffers when writing from the tape to the buffers and scans over buffers with invalid data as well as over buffers which contain out-of-sequence blockettes--this indicates that writing and reading are two separate processes and it is not apparent how this suggests "synchronizing said software buffer index to a hardware buffer index."

In addition, the only condition for advancing the software buffer index in the claimed invention is that it points to a buffer containing processed data and the only condition for resetting the software buffer index is that it has reached a buffer with unprocessed data. Lounsbury does not teach that the read buffer index pointer is advanced just because it points to invalid data or that the read index is stopped when it has reached valid data. Instead, Lounsbury describes a continuous process of reading from the buffers where the read index pointer is continuously advanced regardless of whether the buffer was marked as valid or invalid and a continuous process or writing to the buffers where the write index pointer is continuously advanced to search for the next empty buffer. Again, Lounsbury does not describe any synchronization between the write and read index pointers.

Appellant argues that Lounsbury skips over "valid" data if it does not match the right sequence number, and therefore, the rejection creates a system which would skip over unprocessed data in a buffer rather than sequentially processing it. Br. 11-12. The Examiner responds that this "skipping over" teaching of Lounsbury is not relied upon. Ans. 14. Nevertheless, the references must be considered as a whole and it is improper to rely on only so much of the reference as supports the rejection and to ignore teachings which complicate the analysis. The fact that Lounsbury skips over buffers during reading and writing indicates that it is not doing the same thing as recited in the claims.

The different operation of Lounsbury, which relies on searching the buffers based on valid and invalid indicators and sequence numbers rather than processed and unprocessed data, as claimed, and the fact that no synchronization or coordination is described between the write and read pointers in Lounsbury, provides no reason why one skilled in the art would have been motivated to modify the AAPA to produce a system for "synchronizing said software buffer index to a hardware buffer index" by sequentially searching the buffers and resetting the software buffer index to the next available buffer having processed data. Therefore, we agree with Appellant that Lounsbury does not teach or suggest modifying the AAPA to produce the claimed invention.

CONCLUSION

Appellant has shown that the Examiner erred in concluding that Lounsbury would have taught or suggested modifying the AAPA to provide "synchronizing said software buffer index to a hardware buffer index" by sequentially searching the buffers and resetting the software buffer index to the next available buffer having processed data as recited in claim 1. Accordingly, the rejection of claims 1, 2, 4-17, and 19-24 under 35 U.S.C. § 103(a) is reversed. Cromer has not been stated to address the missing limitation of Lounsbury; thus, the rejection of claims 3 and 18 is reversed.

REVERSED

erc

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
3404 E. Harmony Road
Mail Stop 35
FORT COLLINS, CO 80528